ECEN 4638

Controls Lab 4

Austin Glaser

Jay Greco

1 - Open-Loop Control

To control our system open-loop, we commanded a positive and negative pulse. We empirically determined the width and amplitude of both pulses -- interestingly, we achieved the best results for two non-identical pulses, suggesting that our system has some non-linearities and illustrating (along with many other factors) the problems with open-loop control.

After determining a control signal which produced satisfactory results (see Figure 1.1 for the plot of our accepted trial run), we ran the system 10 more times. Each time, we recorded the settling location for the first and second sections, which should be 3.14 and 0 respectively (see Table 1.2). We experienced an average error of -1.025 radians (33%).

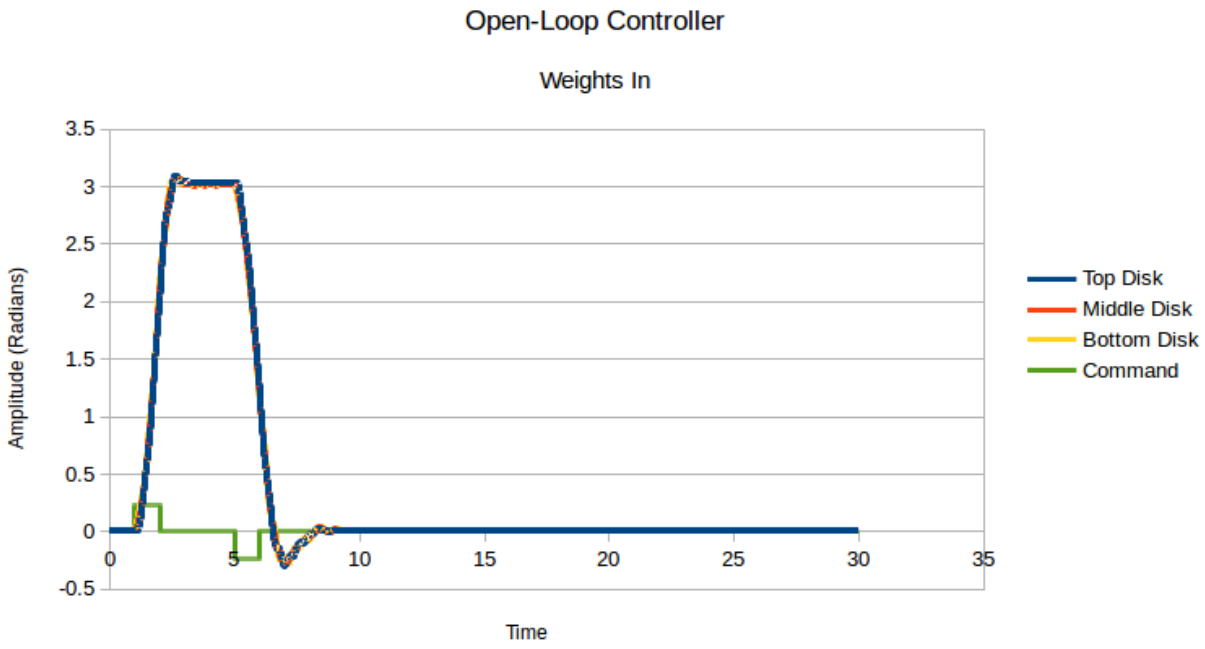

Figure 1.1

Trial	*Location 1*	*Error (from π)*	*Location 2*	*Error (from 0)*
1	2.37	-0.77 (25%)	-0.011	-0.011
2	2.36	-0.78 (25%)	0.043	0.043
3	2.32	-0.82 (26%)	0.050	0.050
4	2.31	-0.83 (26%)	0.265	0.265
5	2.04	-1.10 (35%)	0.001	0.001

6	1.99	-1.15 (37%)	0.005	0.005
7	2.00	-1.14 (36%)	0.078	0.078
8	1.86	-1.28 (41%)	-0.21	-0.21
9	2.21	-0.93 (30%)	0.483	0.483
10	1.69	-1.45 (46%)	-0.46	-0.46

Table 1.2

Next, we used the same control scheme but changed our plant by moving the weights out, thus increasing the moment of inertia of all three disks. As one would expect, the open-loop gain was unable to compensate for the change, resulting in a response even more unsatisfactory (were that possible) than our earlier results. (See Figure 1.3 and Table 1.4). We ended up with an average error of -2.18 radians (69%)

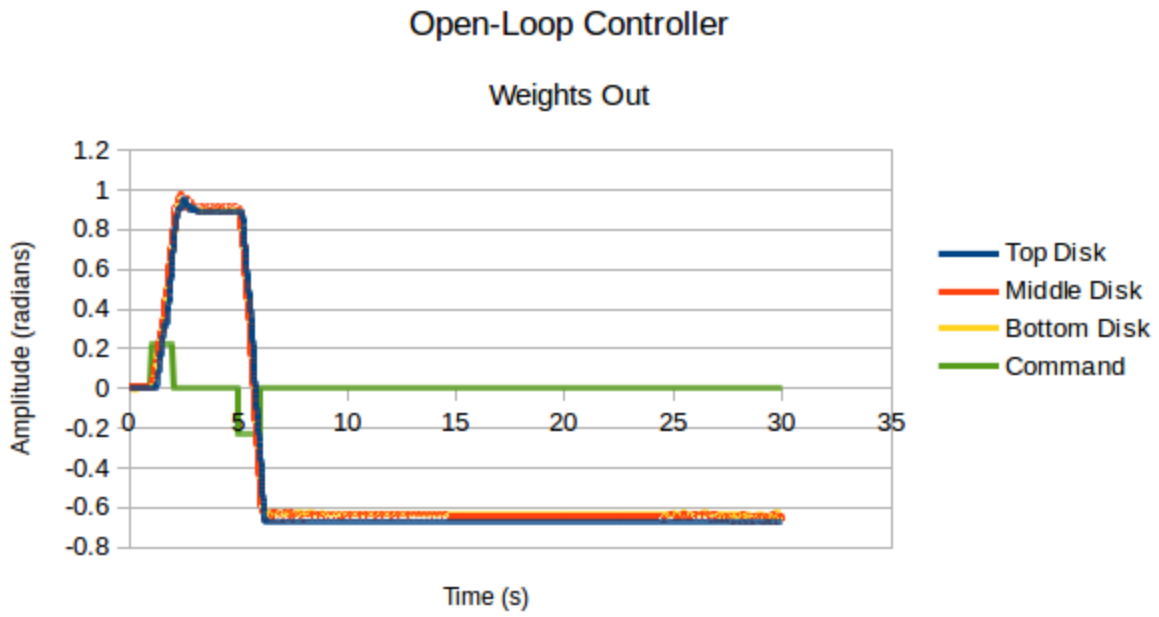

Figure 1.3

Trial	*Location 1*	*Error (from π)*	*Location 2*	*Error (from 0)*
1	0.86	-2.28 (73%)	-0.62	-0.62
2	1.01	-2.13 (68%)	-0.32	-0.32
3	0.80	-2.34 (74%)	-0.61	-0.61

4	0.81	-2.33 (74%)	-0.71	-0.71
5	0.95	-2.19 (69%)	-0.54	-0.54
6	1.20	-1.94 (62%)	-0.20	-0.20
7	1.50	-1.64 (52%)	-0.31	-0.31
8	0.86	-2.28 (73%)	-0.66	-0.66
9	0.78	-2.36 (75%)	-0.72	-0.72
10	0.89	-2.25 (72%)	-0.51	-0.51

Table 1.4

It is clear at this point that open-loop control is not going to work for our system, due to inconsistencies between trials, our system's nonlinearities, and changes that may impact the physical behavior of the system. While an open-loop controller is not able to compensate for, say, the change in moment of inertia of the disks, a closed-loop controller should be able to.

2 - P, PD, PID Control

We began this part of the lab by implementing a proportional controller. We experimented with different values for Kp, but found ourselves ultimately unable to meet the stated requirements in the lab (that is, 1s rise time, 30% overshoot, 3s settling and 2 degree steady-state error). We suspect that part of this is due to the fact that our system has relatively high (and non-linear, especially when almost stopped) damping due to a bearing which is near to failure. (See Figure 2.1 for a plot of our step response). Furthermore, it is an accepted fact that proportional-only controllers are able to achieve fast rise time and decent steady state error, while simultaneously causing high overshoot.

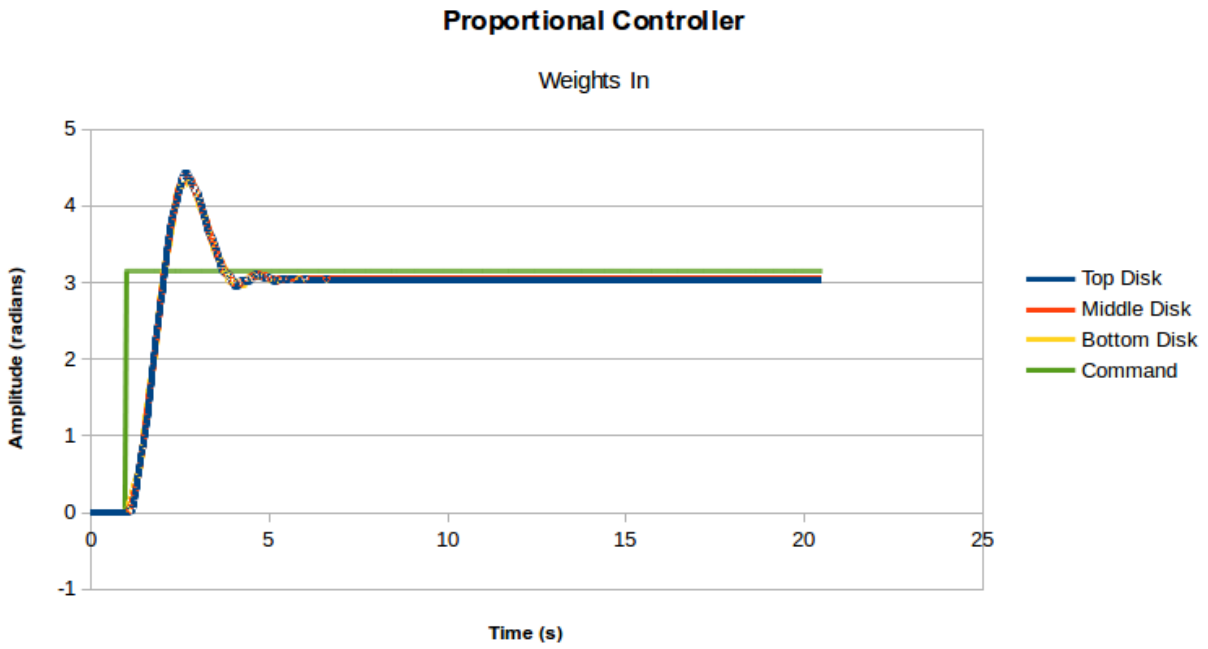

Figure 2.1

Next, we implemented a PD controller (see Figure 2.2). We managed here to improve our performance with respect to rise-time and overshoot, but still encountered problems with steady state errors. This is to be expected, given that the term which corrects for steady-state error is the I term, which is missing from our system. In steady state, the differential is definitionally zero, and if our Kp is large enough to overcome non-linearities in the system we will get unfavorable performance with respect to overshoot, and may even drive the system unstable or hit the physical driver's current limit.

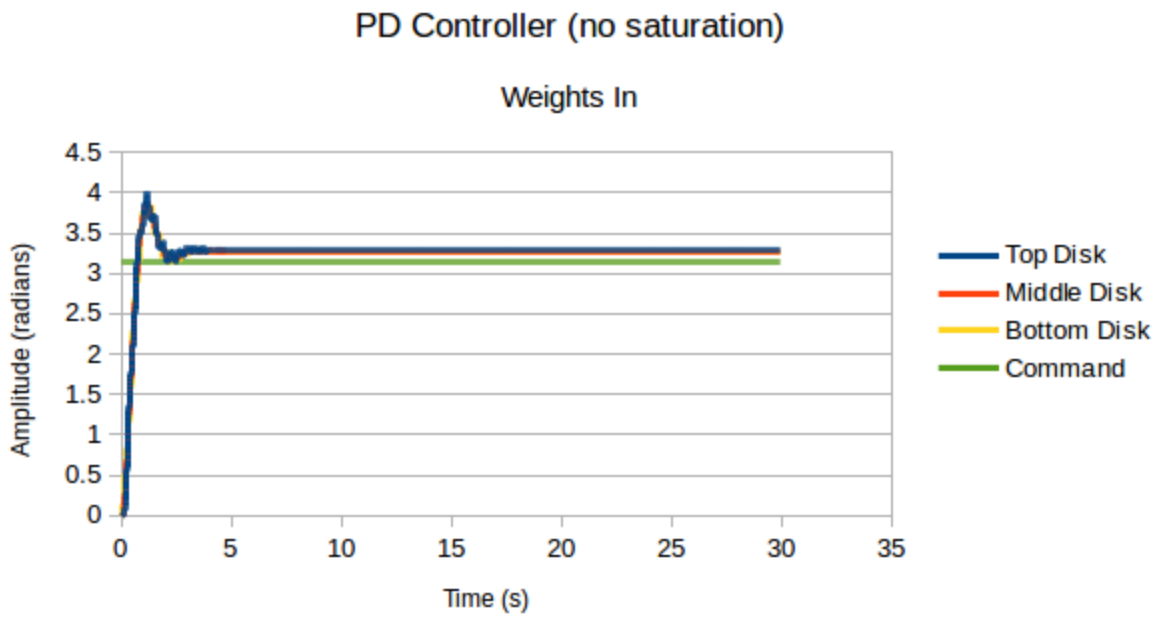

Figure 2.2

When we implemented a PID loop (see Figure 2.3), we saw marginally increased performance. One of our issues was that if we increased our integral term (to reduce steady-state error), we rapidly started encountering situations where our controller would hit the hardware current limit on the motor driver. To combat this, we introduced a saturation value on the output of our controller. This helped us greatly, because it allowed us to tune the PID constants on our loop aggressively without fear of hitting the current limit or damaging our system. (See Figure 2.4 for a plot of our step response with a saturated PID controller).

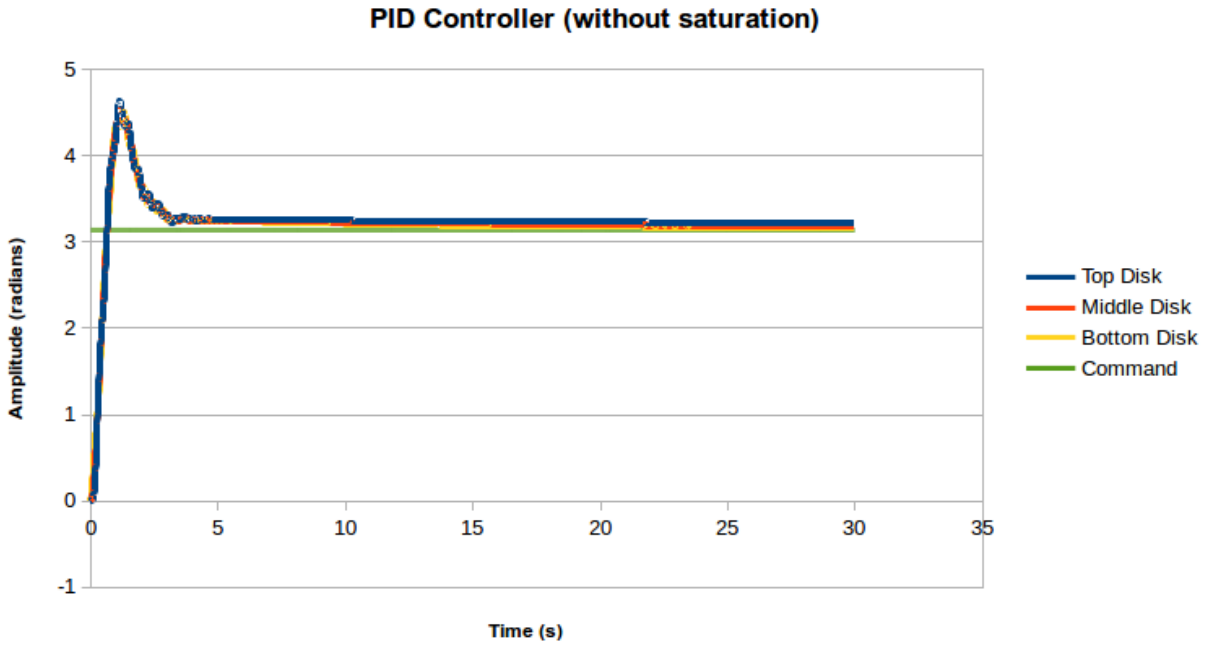

Figure 2.3

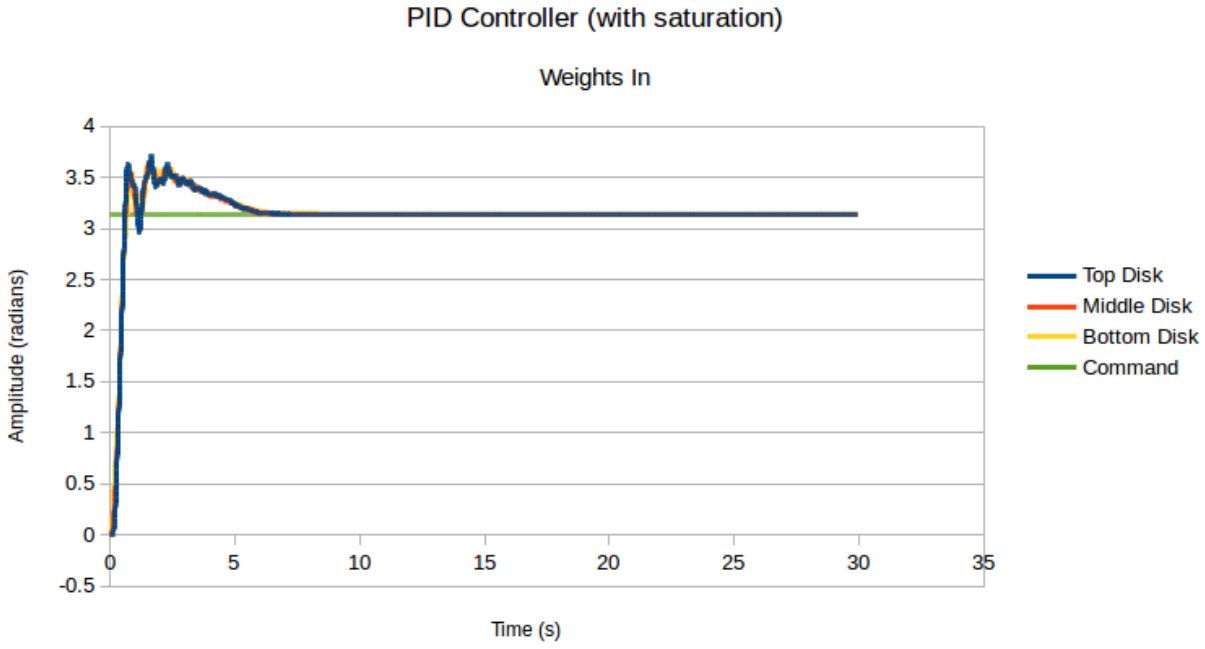

Figure 2.4

Next, we moved the weights outward, changing the plant that we initially tuned the controller to. One of the reasons to used closed-loop control is that it can compensate for these sorts of changes, though performance might be changed or degraded. (See Figure 2.5 for a plot of

our response). These plots are nearly identical, illustrating the fact that our controller is relatively robust to these sorts of variations.

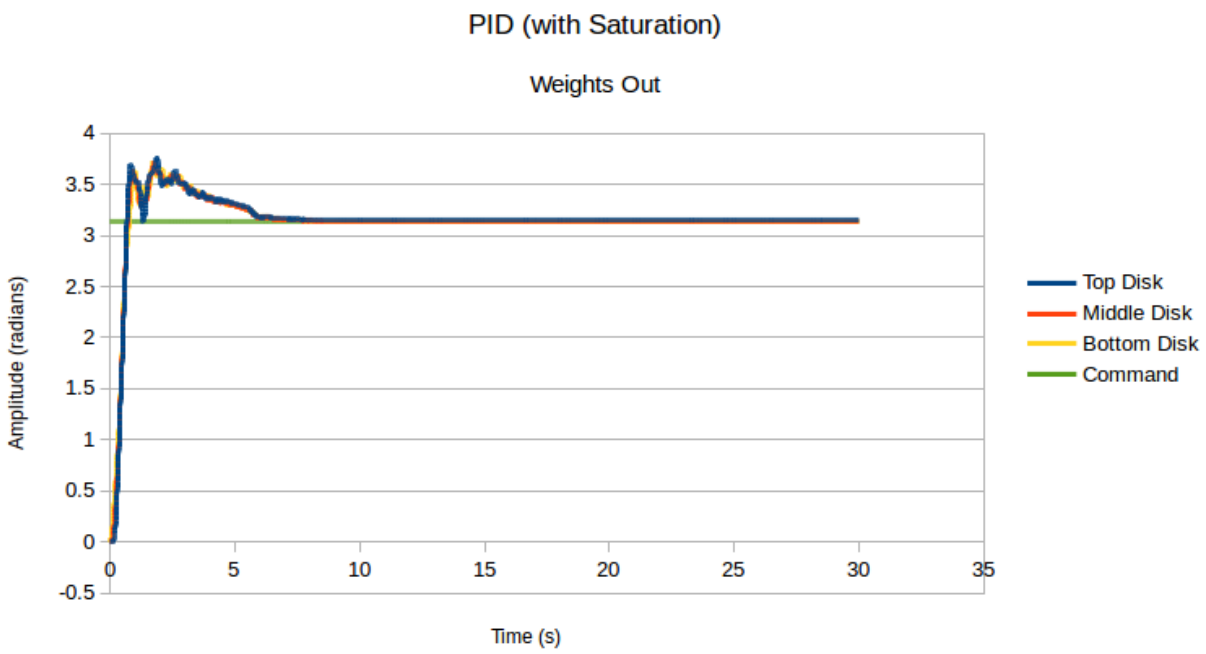

Figure 2.5

The final task was to change our output to the top disk. This is when we started seeing instabilities in our system, and we found it much more difficult to implement a controller satisfying the given requirements. We suspect this is due to the low spring-constant of the rods, which does interesting things to a torque input on the bottom disk. After tuning, we saw marginally degraded performance compared to our bottom-disk controller. (See Figure 2.6).

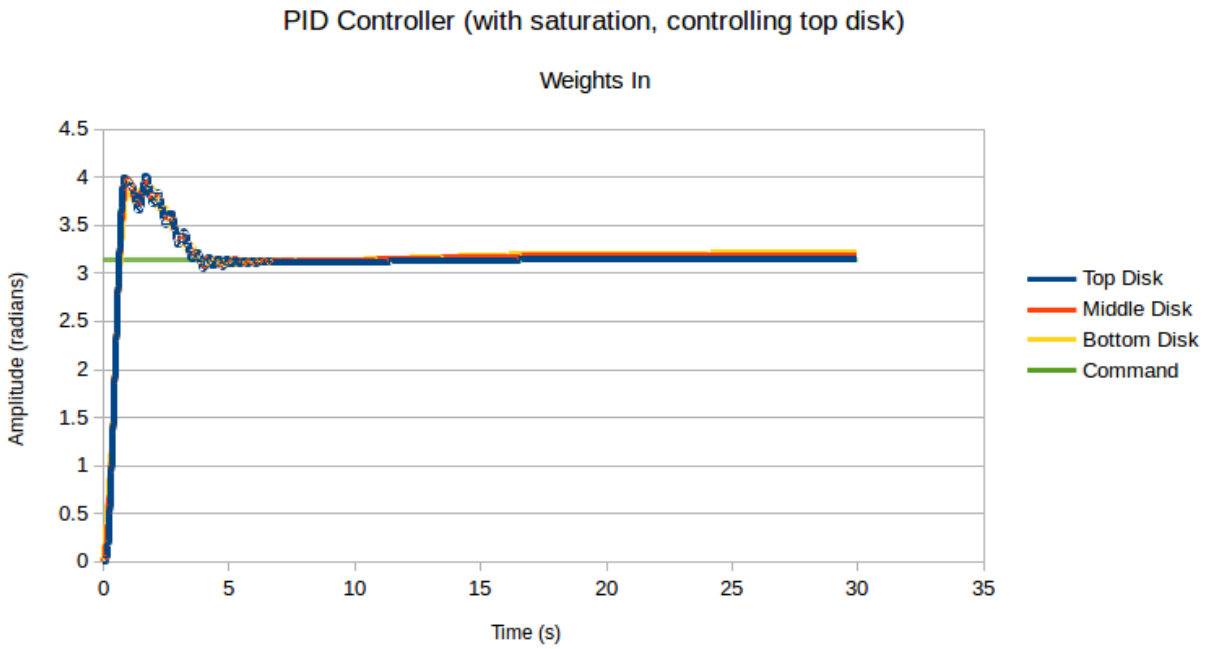
PID Controller (with saturation, controlling top disk)
Weights In
Amplitude (radians)
Time (s)
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
5
10
15
20
25
30
35
Top Disk
Middle Disk
Bottom Disk
Command

Figure 2.6

3 - Simulation

The controller design was simulated in MATLAB using the transfer function model found in labs 2 and 3. A PID controller was applied to our TDS system model. The same proportional, integral, and derivative gains were used as the physical system. The results of the simulations are shown below, in figures 3.1 through 3.3.

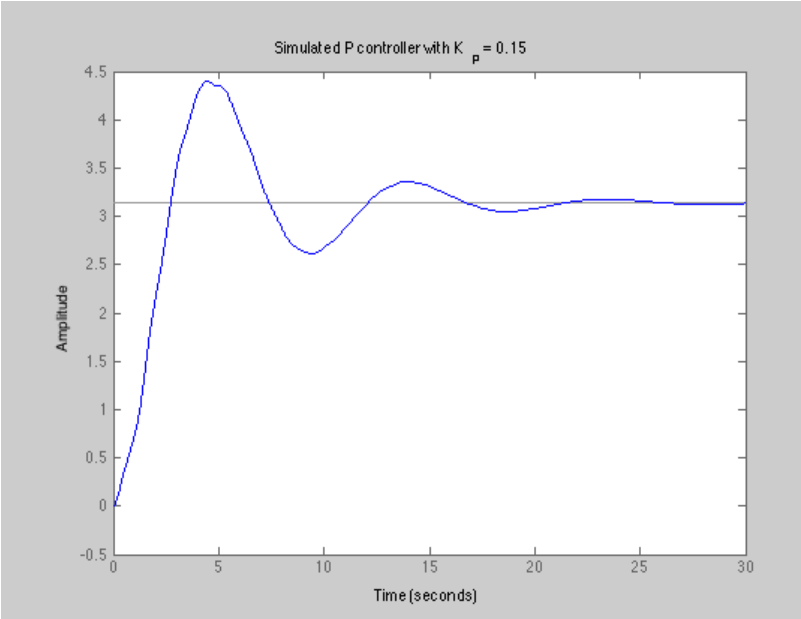

Figure 3.1: Simulated closed loop response using P only controller.

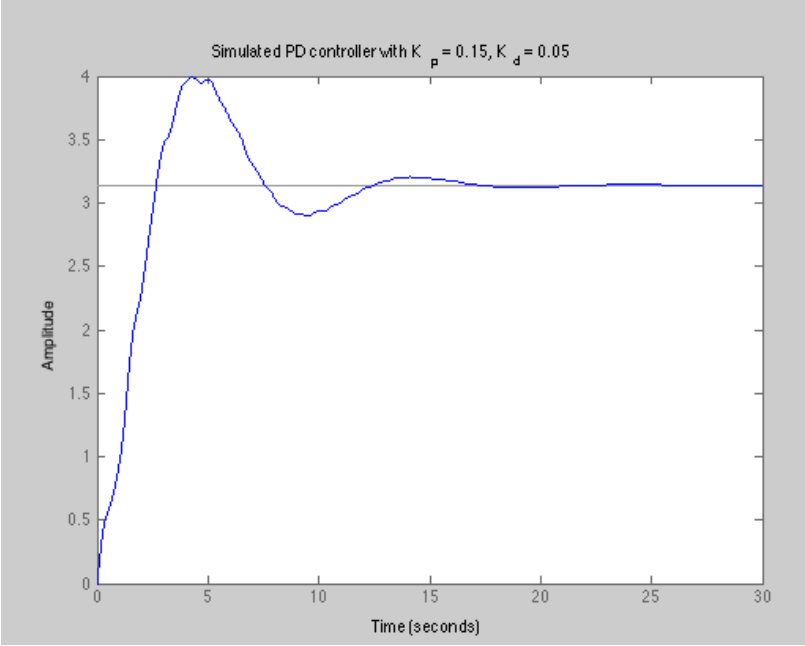

Figure 3.2: Simulated closed loop response using PD controller.

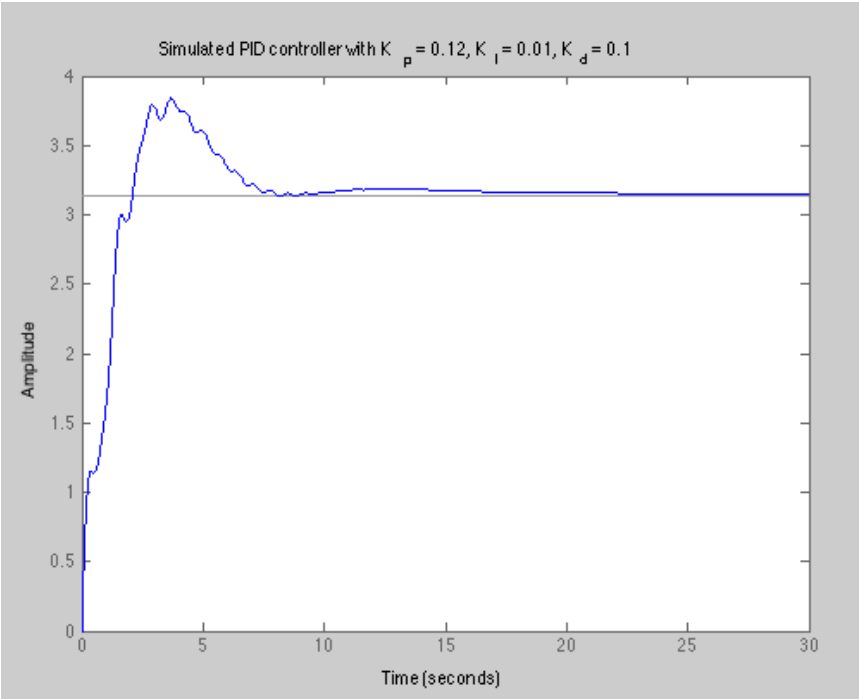

Figure 3.3: Simulated closed loop response using full PID controller.

From the figures, it can be seen that the simulations are close to the experimental data for each controller type. However, the nonlinearities in our TDS system (due to the top disc bearing) have a noticeable negative impact on our system's performance. Mainly, the system tends to settle into specific positions and doesn't easily leave said positions. Overall, however, the simulations can still help us improve upon our controller design, as there are no harmful repercussions of choosing incorrect gains. In this way, it's easier to design a controller without fear of damaging the system or exceeding current limits, etc. Although the system behaves differently than the simulations, the simulations are still great for obtaining an order-of-magnitude and general feel for controller parameters.

This simulation can allow for improvement on our PID design, due to the very small wait time it takes MATLAB to simulate. Changing a constant in LabVIEW, rerunning, and gathering data can take upwards of a few minutes, while the MATLAB simulation can be run in just a few seconds. The numbers were tweaked and the response was observed until the system looked to be behaving better than the original PID controller. The results are included below in figure 3.4.

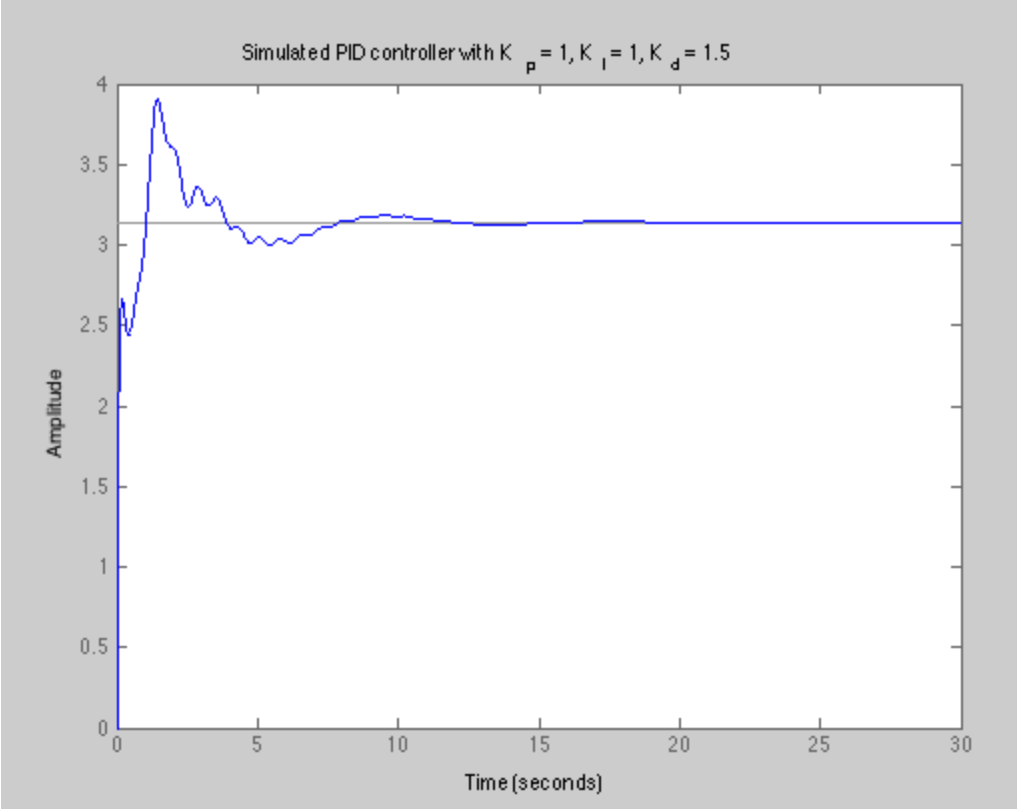

Figure 3.4. A better tuned system obtained through MATLAB simulation.

It's worth noting that there is a downside to a MATLAB simulation: It does not model certain details of the system, such as maximum controller output. In this case, the simulated TDS response has both a faster rise time and a smaller maximum overshoot than the controller that was initially designed. However, it's very possible that the physical system cannot safely support gains as high as the Kp, Ki, and Kd gains chosen for the simulation. It is likely that the FPGA controller would either saturate or hit the current limit for these high gains. It's clear that the best design methodology for our TDS uses a balance of both simulation and physical testing. This speeds up the design process, but also keeps in place checks to ensure that any results obtained are not outside the reasonable physical boundaries of the system.

Also noted is that the system is easily destabilized by choosing "bad" gains. We experienced this issue when we changed our system output from the bottom disk (disk 1) to the top disk (disk 3). Our derivative gain was much too high, and, as a result, the system became unstable. This issue can be reproduced in the MATLAB simulations. Changing the system's output to disk 3 in MATLAB and setting the same gains as the physical system, the simulation clearly demonstrates instability. Figure 3.5 shows the relevant MATLAB simulation.

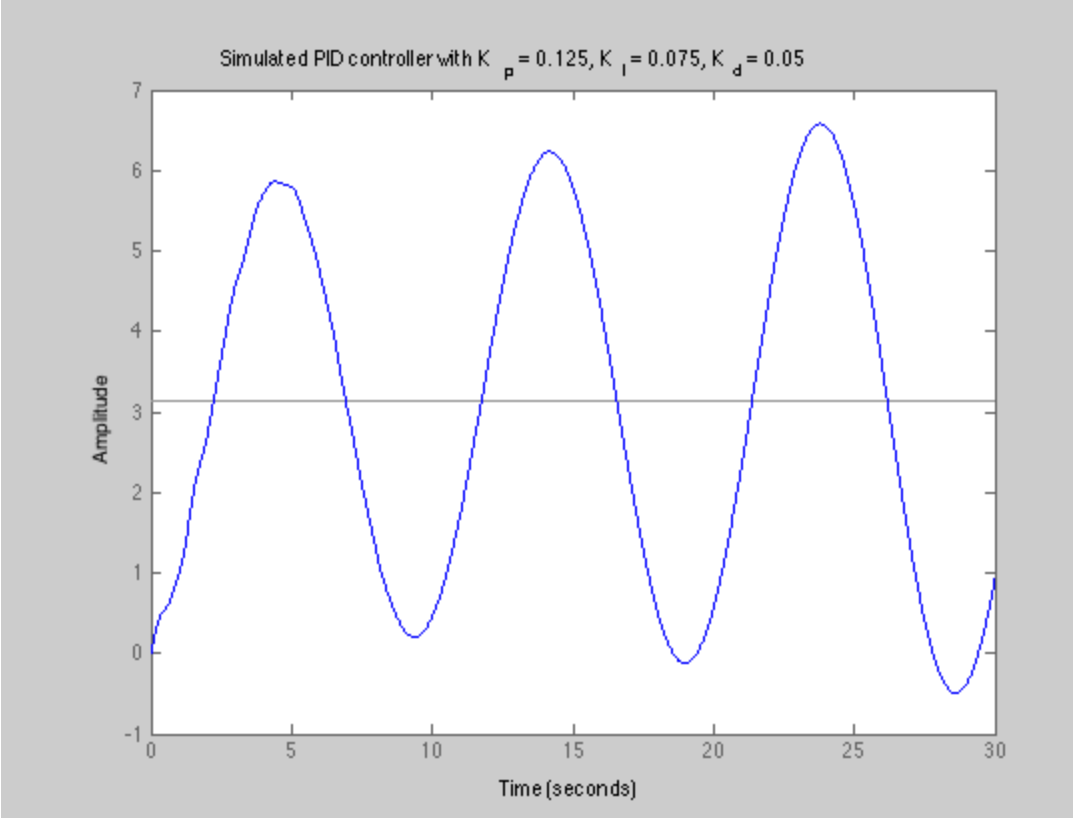

Figure 3.5: Simulated instability of the TDS due to incorrect gains.

Using MATLAB's root locus analysis, it can be seen that this issue can arise due to an integral gain that is too large. Figure 3.6 shows the root locus of the simulated system with an integral term that is an order of magnitude higher than both the proportional and derivative term. For a specific range of k, the system's poles move into the right half plane before arriving at their corresponding zeros.

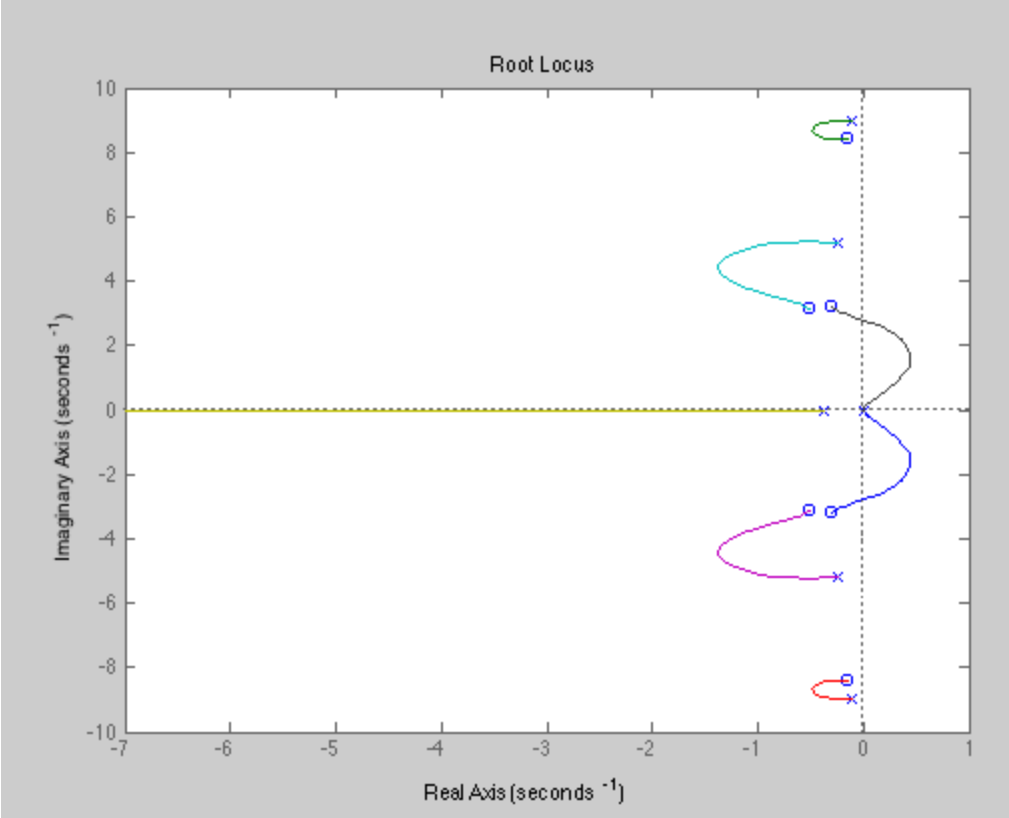

Figure 3.6: Simulated system with very high integral term. Note the RHP poles.

In contrast, changing the proportional gain or derivative gains to be much higher than the others creates no such instability, and this can be seen in their corresponding root loci. Figures 3.7 and 3.8 detail these root loci.

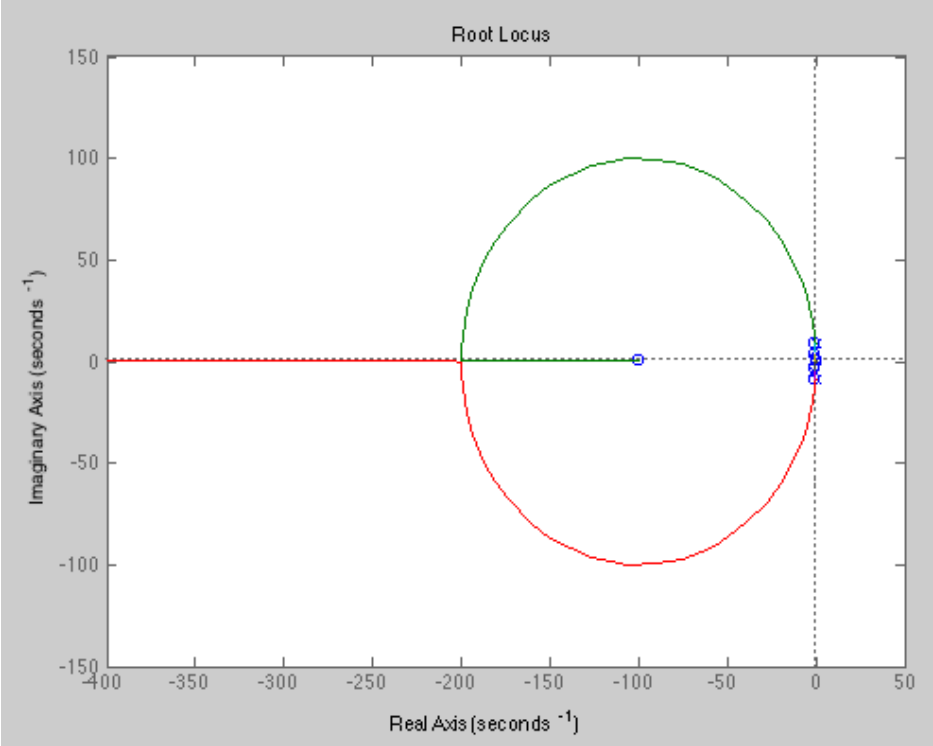

Figure 3.7: Root locus of TDS with very high proportional gain. Note that there are no RHP poles.

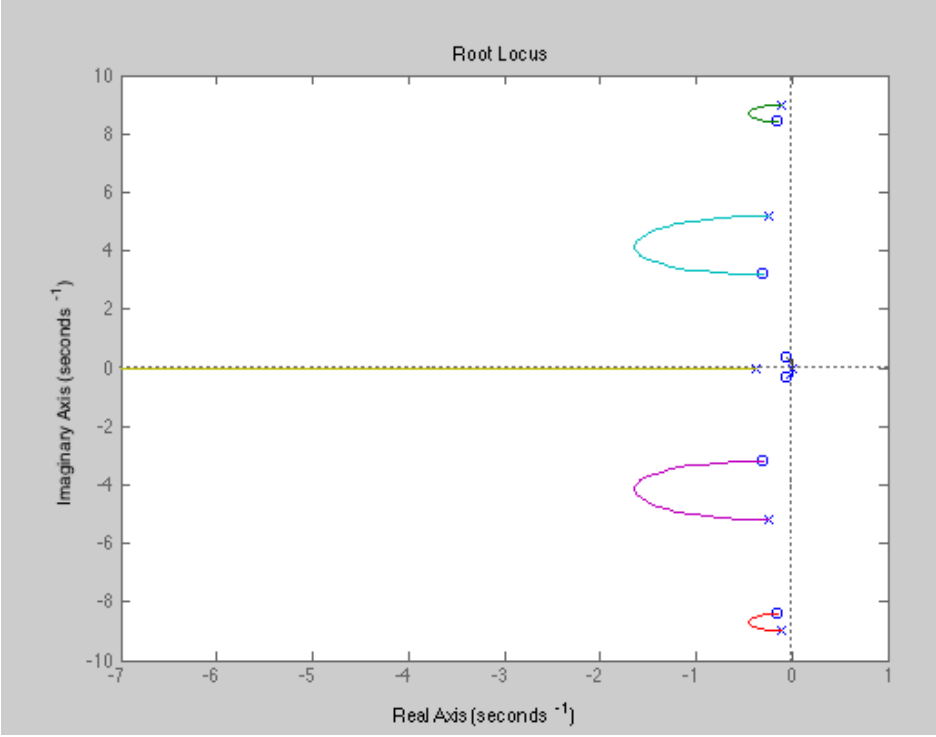

Figure 3.8: Root locus of TDS with very high derivative gain. Note that once again there are no RHP poles.

We can explain our system's instability with the root locus plots that were generated with MATLAB. It's obvious that the integral term was much too high for the system, and, because of the chosen gain, the double pole at the origin moved into the right half plane. In theory, the system would become stable with an high gain (the poles move back into the left half plane), but this isn't possible with our hardware. The gains are simply limited by the voltage and current our controller can supply without saturating or hitting the current limit.

When controlling the top disc, the system becomes even more sensitive to the various gains. Figure 3.9 shows the root locus of the system, but this time controlling the top disc as opposed to the bottom disk. It can be seen that there is a very small range of gain values for which there are no RHP poles; choosing the wrong gain can easily push the system to be unstable. The complex poles are only in the left half plane for very small values of k, and move to the right half plane with any sizeable gain increase.